Empirical Exploration of Clustered Federated Learning

SIDDHARTHA BHATTACHARYA, Michigan State University, USA
ZANE ARIDI, Michigan State University, USA

Federated learning (FL) requires each client device in a collaborative training network to train a local model using on-device training data for some iterations, then transmitting the model updates to a central server, which will aggregate received updates and send back the global model. However, clients in the network may have local training data generated from different statistical distributions, violating the common assumption of independently and identically distributed (iid) data in machine learning. Thus, we consider a network of clients with different distributions of training data (non-iid), with the objective of minimizing the loss function of each client within their respective domains. We evaluate the ability of state-of-the-art unsupervised clustering algorithms to correctly cluster clients based on their model parameters and to converge the global models within each cluster.

ACM Reference Format:
Siddhartha Bhattacharya and Zane Aridi. 2018. Empirical Exploration of Clustered Federated Learning. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Federated learning aims to learn a single global model through decentralized training and centralized model aggregation. Specifically, each client trains a model on their local data for some predetermined number of iterations, then shares the model with a central server, which aggregates all received models into a new global model [7]. This aggregation function typically involves taking the average value of each parameter in the received client models, or Federated Averaging (FedAvg).

This process assumes that training data across clients is sampled from the same data distribution, in an independently and identically distributed manner (i.i.d.), which is often not the case. Real-world training networks often have clients with heterogeneous data distributions [13]. For example, a network of autonomous vehicles may aim to train a shared road-sign detection model, but road signs from one state may look very different from those from another state. The global model trained on non-i.i.d. data may perform well on average, but achieve poor performance for each individual client [9, 12].

When training machine learning models in a decentralized paradigm, relaxing the assumption of i.i.d. training and testing data typically comes at the cost of local model performance [4]. Federated learning is especially prone to this, as there is no centralized control over the training data at each client device, as compared to traditional distributed training.

In a federated learning network performing supervised learning, each client has a dataset, D_i consisting of samples of the form $(\mathbf{x}, y)$ where $\mathbf{x}$ denotes the observed features and y denotes its

Authors' Contact Information: Siddhartha Bhattacharya, Michigan State University, East Lansing, USA, bhatta70@msu.edu; Zane Aridi, Michigan State University, East Lansing, USA, aridizan@msu.edu.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

corresponding label. These samples for client k are generated from some probability distribution $P_k(\mathbf{x}, y)$ [13]. Heterogeneity in this setup implies that $P_i \neq P_j$ for some clients i, j.

There are a number of ways that heterogeneity can arise and affect the learning process in federated learning, which can broadly be categorized into two types: feature skew and label skew [13]. Feature skew refers to the scenario in which two clients j, k may have training samples $(\mathbf{x_k}, y_k) \sim P_k$, $\mathbf{x_j}, y_j \sim P_j$, respectively, where $y_k = y_j$ but x_j may be very different from x_j. Continuing with the road sign example, if the road signs in two states look very different but have the same meaning or label, then this is an example of feature skew.

Label skew refers to the case where $\mathbf{x_k} = \mathbf{x_j}$ or are highly similar, but $y_k \neq y_j$. Intuitively, this may be the case when different clients have differing opinions on the label of a sample.

2 Related Work

A number of approaches have attempted to address the issue of non-i.i.d. clients. Some aim to learn a global model which responds well to fine-tuning on each client device [1, 11], while others have suggested performing federated learning only on a subset of the layers within the model [5].

Another line of research attempts to cluster clients into groups, where each cluster contains clients who have training data from similar distributions [2, 3, 6, 8].

[3] presents a system-level clustering method called iterative federated clustering algorithm (IFCA). This algorithm presents a method for privacy-preserving clustering of client devices. It interleaves the federated learning process with updating cluster estimates of each client. [2] expands upon this, by creating a one-shot clustering framework and interleaving updates. We expand upon this work and study the performance of a one-shot federated clustering approach with no interleaving updates.

3 Problem Statement

We consider a clustered federated learning network, where we aim to learn k cluster models. Each cluster aims to minimize the average of the loss function on the nodes belonging to the cluster. Therefore, we implemented a decentralized training setup with multiple central aggregation steps. Specifically, clients from across different clusters may be trained in parallel, but model aggregation will only be performed among clients from the same cluster.

Given N clients, let $\mathcal{D}_i = \{(x_{ij}, y_{ij})\}_{j=1}^{n_i}$ represent the local dataset of client i, where $x_{ij} \in \mathcal{X}$ is an input sample, y_{ij} is the corresponding label, and n_i is the size of the clients' dataset. Each sample $(x_{ij}, y_{ij}$ is generated from a probability distribution P_i. Then, the global goal of federated learning is to minimize the empirical risk:

$$\min_{w \in \mathcal{W}} \sum_{i=1}^{N} \mathcal{L}_i(w; x_i),$$

where $\mathcal{L}_i(w) = \frac{1}{n_i} \sum_{j=1}^{n_i} \ell(f_w(x_{ij}), y_{ij})$ is the local empirical risk for client i, $\ell(\cdot, \cdot)$ is a loss function, and w represents the model parameters.

However, when clients' data distributions $P_i(x, y)$ vary significantly, a single global model f_w may fail to generalize well across all clients. To mitigate this, we propose clustering the clients into groups with similar underlying distributions, thereby enabling the learning of group-specific models optimized for each cluster.

In clustered federated learning, the objective is to partition the N clients into K clusters, where each cluster k is associated with a model w_k. Each client computes its empirical risk using the model corresponding to its assigned cluster. Let C_k denote the set of clients assigned to cluster k, and the objective becomes:

$$\min_{\{w_k\}_{k=1}^{K}} \sum_{k=1}^{K} \sum_{i \in C_k} \mathcal{L}_i(w_k),$$

where $\mathcal{L}_i(w_k) = \frac{1}{n_i} \sum_{j=1}^{n_i} \ell(f_{w_k}(x_{ij}), y_{ij})$ is the empirical risk for client i using the model w_k of its assigned cluster. The optimization seeks to minimize the overall empirical risk by learning corresponding cluster models. This formulation rests on accurate cluster predictions by the clustering algorithm.

Thus, we study the ability of the K-means clustering algorithm to correctly cluster clients into their data distributions. We perform an empirical evaluation on both the cluster predictions as well as the final cluster models obtained from performing clustered federated learning.

Assumptions:

- $P_i \cap P_j = \emptyset$ for $i \neq j$, or that the data distributions are non-overlapping.
- We have full participation of clients in the federated learning process.

4 Methodology

4.1 Dataset Selection

Our model's performance is tested using MNIST, CIFAR10, and CIFAR100. All three datasets are using an 86 to 14 train/text split. We denote train size as N and test size as M. Normalization transforms are applied to all loaded data to make pixel values between -1 to 1.

4.2 Training

We started with a lightweight model to evaluate CIFAR10 and MNIST datasets. This allowed us to have quick experimental iterations. We used 2 conv layers and 3 linear layers. When evaluating CIFAR100 we needed a more robust model so we used 6 conv layers and 2 linear layers.

Each client trains their local models using a learning rate of 0.01, cross-entropy loss, and mini-batch-SGD optimization with a batch size of 32.

Specific model details can be found in the source code (click here).

We consider federated learning networks with 5 clients per cluster. For experiments with 2 clusters, we have 10 clients, and for experiments with 5 clusters, we have 25 clients.

The network is trained for 50 rounds and the client trains for 1 local epoch within each round before sending their model to the server for aggregation.

Parameters to control each of the training configurations, cluster sizes, datasets, and heterogeneity are specified in the config.yaml file.

4.3 Cluster Creation

Let there be n nodes in the network belonging to one of K disjoint data distributions (true clusters), denoted by the true clustering map $C^*[n] \mapsto i \in \{1, \ldots, K\}$. The nodes belonging to cluster i generate training samples from some probability distribution, $x \sim P_i$, where $P_1, \ldots, P_k$ are non-overlapping distributions, i.e., we assume that each cluster has a unique distribution of training data that does not intersect with other clusters.

Clients can either be evenly or unevenly distributed among the clusters. This setting is toggled by a parameter in our experiment configuration file. In the the case of even distribution,

$$|C_1| = \ldots = |C_k| \tag{1}$$

where $|C_i|$ denotes the number of clients belonging to cluster i.

We also consider a more practical setting, where we cannot guarantee an equal assignment of clients to each cluster. This situation may arise in edge-device networks, where the density of

clients varies by geographic location. To construct uneven cluster sizes, we assign clients to clusters by weighted probabilities. We calculate the probability of a client being assigned to the i-th cluster to be proportional to the inverse of a harmonic sum. Specifically,

$$|C_i| = \left\lfloor \frac{n}{(i+1) \cdot \sum_{j=0}^{k-1} \frac{1}{j+1}} \right\rfloor \tag{2}$$

In effect, $|C_0| > |C_1| > \cdots > |C_k|$.

Recall that clusters are defined by unique data distributions. Following related works, we utilize 2 artificial methods of heterogeneous cluster creation: **(i)** Domain Shift (feature skew) **(ii)** Label Heterogeneity (label skew).
(i) To perform the domain shift rotation transform is used. The selected angle, denoted r_i for cluster i, is

$$r_i = \left(\frac{i}{k+1}\right) \times 360 \tag{3}$$

where k is the number of clusters. This leads to each client having

$$c_n = \frac{N}{k_c}, c_m = \frac{M}{k_c} \tag{4}$$

data-points, where c_n and c_m are the quantity by train and test respectively, and where k_c is the number of clients per cluster.
(ii) To perform label heterogeneity the number of classes assigned to each cluster is

$$k_n = n/k \tag{5}$$

where n is the number of classes. This leads to each client having

$$c_n = \frac{(\frac{N}{n} * k_n)}{k_c}, c_m = \frac{(\frac{M}{n} * k_n)}{k_c} \tag{6}$$

data-points.

4.4 Cluster Algorithms

We use the K-means algorithm to perform clustering. Each centroid is a client's weight mappings by layer. We implemented 2 distance metrics, Point-wise Manhattan distance and Layer Distance. K-means selects random clients as the initial centroids, and will then cluster all other points to their respective closest centroids.

$$\underset{C}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{x \in C_i} \operatorname{dist}(x, j) \tag{7}$$

We want to minimize the sum of the distances between each client in the cluster, x, and the cluster's centroid j. Since our data points are neural networks themselves, we consider different methods for computing the distance between selected centroids and other clients' models.

4.4.1 Normalization. We started writing an algorithm for normalization to create a standard range for the model weights by subtracting standard deviation and dividing by mean. We realized this would allow for scale invariance which is not necessary for comparing model weights, if anything scale is important in this regard to capture. We soon realized normalization using max and min values may be more useful but did not become a priority for further exploration.

4.4.2 Point-wise Manhattan Distance. We calculate the Manhattan distance between a client model and the selected centroid model. Each client's model is flattened into a vector representation, $W_i \in \mathbb{R}^m$, where m is the number of parameters in the model. Then, we calculate the distance between any two clients j, k using their respective weight vectors, W^j, W^k:

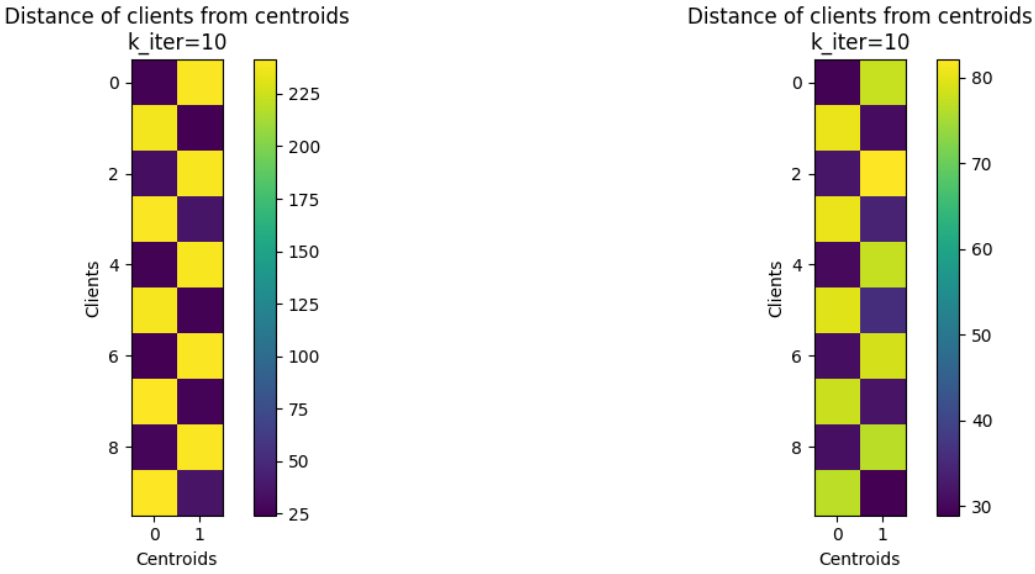
$$\text{dist}(j,k) = \sum_{i=1}^{m} |W_i^j - W_i^k| \tag{8}$$

This is equivalent to the L1 distance between the flattened weight vectors.

4.4.3 Layer Distance. We had thought our K-means algorithm would not be sufficient to cluster the data due to a problem called permutation equivariance. This would occur as different models learn features at different kernels within their convolutional or linear layers while still representing the same underlying function. For example, the right edge for one class may be learned in kernel 1 of conv1 in one model versus kernel 3 of conv1 in another model. We thus calculated the distance between each kernel of conv1 between clients and centroids to find the minimum kernel-wise Manhattan distance between 2 models.

$$\text{dist}(j,k) = \min_{j,k} \sum_{l=1}^{L} \left| W_{j,l} - W_{k,l} \right| \tag{9}$$

Where $W_{i,l}$: Weights of kernel l in the convolutional layer of model i. $W_{j,l}$: Weights of kernel l in the convolutional layer of model j. L : Total number of kernels in the convolutional layer. While the layer-distance metric is functional in the code base, we decided not to test layer distance once we ran t-SNE and figured there may be a bug in our K-means code.



(a) MNIST

(b) CIFAR-10

Fig. 1. Distance matrix of 10 clients from centroids, after training for 1 epoch and running K-Means for 10 iterations using L1 distance. The clients were split into two even clusters, one with the original MNIST or CIFAR-10 dataset, and the other with a rotated version.

5 Experimental Evaluation

5.1 Weighted Purity Metric

To evaluate the accuracy of cluster predictions, we develop a weighted purity metric. For n nodes in the network belonging to one of K disjoint clusters, denoted by the true clustering map $C^*[n] \mapsto i \in \{1, \ldots, K\}$. Recall that $\hat{C}$ denotes the predicted cluster map and $C_1, C_2, \ldots, C_K$ be the cluster sets, where each $C_i = \{n : \hat{C}(n) = i\}$.

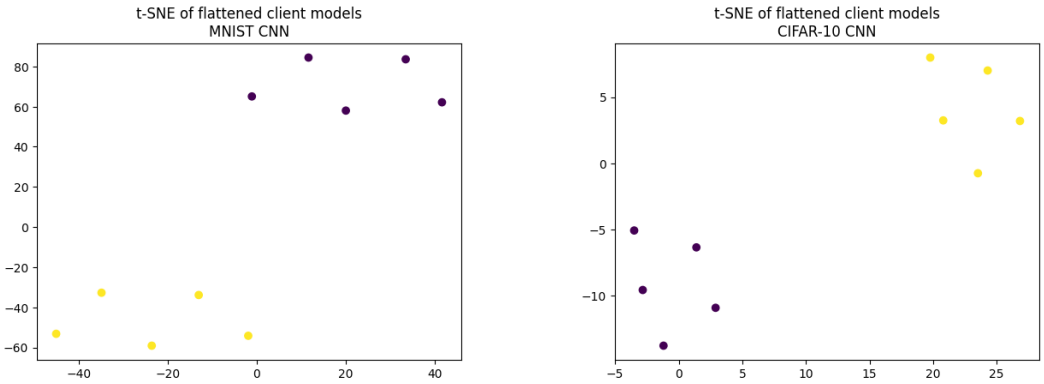
We propose a weighted in-cluster purity metric that quantifies the degree to which predicted clusters contain nodes from the same true cluster.

The in-cluster purity is defined as

$$IC(C_i) = \frac{|\{\{i, j\} \mid i, j \in C_i \text{ and } C^*(i) = C^*(j)\}|}{\binom{|C_i|}{2}} \tag{10}$$

Then our weighted purity metric is defined as

$$A(\{C_1, \ldots, C_K\}) = \sum_{i=1}^{K} \frac{IC(C_i)}{|C_i|} \tag{11}$$

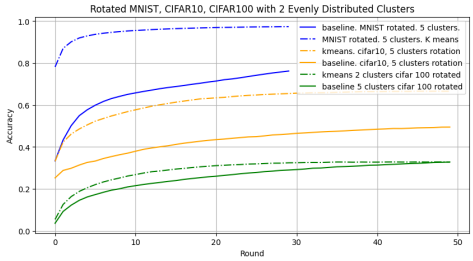


(a) MNIST (b) CIFAR-10

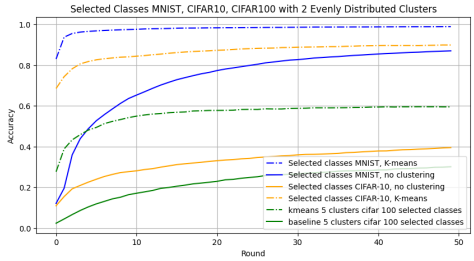
Fig. 2. t-SNE visualization for flattened weights of 10 client models split into 2 clusters, training for 1 epoch. Results for models trained on MNIST and CIFAR-10 are shown in Figures 2a and 2b, respectively.The clear t-SNE separability in both figures indicates that there exists a non-trivial difference in the features learned by each cluster of clients.

5.2 Analysis

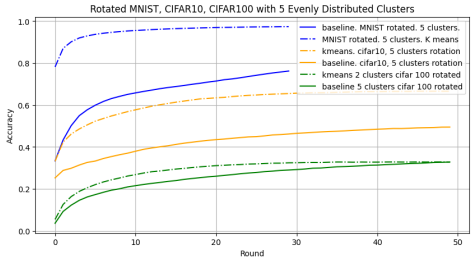
5.2.1 General. Our experimental results comparing baseline federated learning accuracy against our clustered results are, given in 1. Clustered outperformed baseline FedAvg in terms of convergence rate in every scenario. In terms of final accuracy, clustered federated learning outperformed the baseline in all but 2 cases, where we achieved equal performance. Clustered learning also converged to higher accuracy faster in every trial we ran, shown in 3. All results were using Point-wise Manhattan distance.



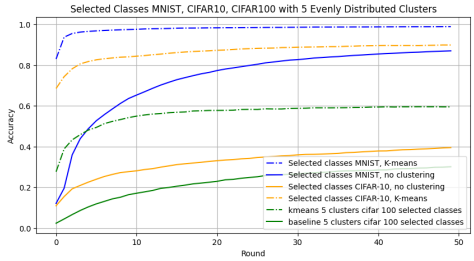
(a) 2 clusters, rotated (feature skew)



(b) 2 clusters, selected classes (label skew)



(c) 5 clusters, rotated (feature skew)



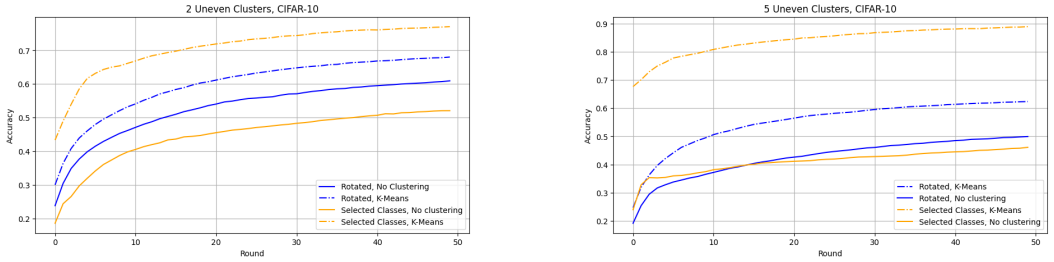
(d) 5 clusters, selected classes (label skew)

Fig. 3. Average client accuracy over 50 rounds of federated learning for clustered and non-clustered (baseline) experiments.

Even vs Uneven	Dataset	# of Clusters	Type of Split	Accuracy Change	Cluster Purity
Even	MNIST	2	Rotation	5%	1.0
Even	CIFAR-10	2	Rotation	5%	1.0
Even	CIFAR-100	2	Rotation	0%	1.0
Even	MNIST	2	Selected Classes	3%	1.0
Even	CIFAR-10	2	Selected Classes	28%	1.0
Even	CIFAR-100	2	Selected Classes	10%	1.0
Even	MNIST	5	Rotation	21%	0.88
Even	CIFAR-10	5	Rotation	17%	1.0
Even	CIFAR-100	5	Rotation	0%	0.38
Even	MNIST	5	Selected Classes	12%	0.88
Even	CIFAR-10	5	Selected Classes	50%	1.0
Even	CIFAR-100	5	Selected Classes	29%	1.0
Uneven	CIFAR-10	2	Rotation	7%	1.0
Uneven	CIFAR-10	5	Rotation	12%	1.0
Uneven	CIFAR-10	2	Selected Classes	25%	1.0
Uneven	CIFAR-10	5	Selected Classes	43%	1.0

Table 1. Accuracy gain and cluster purity for clustered federated learning with different datasets, splits, and settings. Accuracy gain is computed as the difference in final accuracy between the experiment run and the baseline (no clustering). Cluster purity is defined by Equation 11

5.2.2 t-SNE. After having some initial trouble with K-means (that we later found was due to a bug in our code) we decided to apply t-SNE [10] to see if there was a way to reduce the dimensionality



(a) 2 Uneven Clusters

(b) 5 Uneven Clusters

Fig. 4. Average client accuracy over 50 rounds of federated learning CIFAR-10 with unevenly distributed clusters

and visualize the data of our models without losing enough info to determine separability. We trained 10 clients, split into 2 clusters of rotated datasets for 1 epoch. We then flattened all of their weights and performed t-SNE for visualization, shown in 2.

After running t-SNE, we observed that the weights of these flattened models are well-separable and that K-means should work which prompted us to review our code and find the bug.

5.2.3 By Dataset. **(i)** CIFAR-100 gained no performance increase when data had undergone a rotational domain shift. When 5 clusters were applied it also had the lowest cluster purity of any trial. We tried testing this with 10 initial epochs before clustering thinking giving the models more time to converge before clustering may allow for more separability but got the same results, found in the source code under ()experiments/results/cifar100-purity-test. With 2 clusters it still managed to attain perfect cluster purity which seemed to have not benefited the accuracy at all. However, it gained 10% and 29% across our label heterogeneity trials with selected classes. Notably, there was a higher increase with 5 clusters compared to 2. It makes sense that as we increase the amount of clusters in CIFAR100 performance increases as each model has to worry about fewer conflicting classes which allows each model to be more fine-tuned without over-fitting. The lack of performance increase on rotation suggests that the data may not be easily separable. In the future, t-SNE should be run on CIFAR100 with rotational domain shift to determine if the data is as separable as with CIFAR10 and MNIST.
(ii) CIFAR10 had a performance increase across all its trials, most notably with label heterogeneity. While it had modest performance increases in domain shift it is notable that with 5 clusters there was a performance increase of 12% over 2 clusters. For label heterogeneity, there was a significant performance increase of 28% at 2 clusters and 50% at 5. This supports that increasing clusters leads to better performance not only with label heterogeneity but with domain shift as well.
(iii) MNIST had performance increases in all its trials, most notably with domain shift. Its performance increase in selected classes was marginal compared to the CIFAR datasets; however, it performed the best across all datasets in terms of domain shift. This can be visualized against CIFAR10 in Figure 1. This could be because of MNIST's reduced dimensionality leading to better performance with domain shift. It also had the largest factor if increase between 2 and 5 clusters going from 5% to 21% with domain shift, and from 3% to 12% with selected classes. The best increase CIFAR10/100 had was a factor of 3x, while MNIST's was 4x. Interestingly MNIST also has imperfect cluster purity with 5 clusters yet it still performs well unlike when there is imperfect cluster purity with CIFAR100. Would perfect cluster purity have resulted in higher accuracy?
(iv) When comparing datasets it seems that there is a consistent increase in performance due to

more clusters. One limitation that would need to be tested is at what point adding more clusters causes harm to performance. For selected classes, we should test $k > n$ where k is the number of clusters and n is the number of classes. For rotation, we should increase the number of clusters until we can determine at what rotational interval the data is too similar to be separable. Another matter that should be investigated is why CIFAR does better with label heterogeneity compared to MNIST doing better with domain shift.

5.2.4 Even Vs Uneven. Clustered federated learning continued to outperform baseline FedAvg in the case of uneven cluster distributions, as seen in Figures 4a and 4b. This indicates that K-means based clustered federated learning has potential for more practical applications in which cluster sizes are unequal.

6 Conclusions and Future Work

6.1 Conclusions

This study demonstrated the effectiveness of clustered federated learning using the K-Means algorithm with the L1 distance metric. Across multiple datasets and configurations, clustering consistently outperformed baseline Federated Averaging (FedAvg) in terms of accuracy and convergence speed. Key findings include:

- **Improved Performance with Clustering:** Clustered federated learning showed significant improvements in both final accuracy and convergence rates across various datasets. The ability to partition clients based on data distributions addresses the limitations posed by non-i.i.d. data, highlighting the promise of clustering in federated setups.
- **Impact of Cluster Sizes:** Evenly distributed cluster sizes achieved better accuracy compared to uneven configurations, emphasizing the importance of balanced client distribution for clustering algorithms. However, the method demonstrated robustness in practical scenarios with uneven client distributions.
- **Benefits of Increased Clusters:** Larger numbers of clusters generally led to higher accuracy, likely due to better specialization of cluster-specific models. However, this improvement did not always translate to faster convergence, suggesting a trade-off between model specificity and computational overhead.
- **Dataset-Specific Observations:**
  - **MNIST:** Outperformed other datasets in scenarios involving domain shifts, likely due to its lower dimensionality and simpler features.
  - **CIFAR-10 and CIFAR-100:** Showed greater performance gains with label heterogeneity, particularly as the number of clusters increased, underscoring the adaptability of clustered learning to complex and diverse datasets.
- **Cluster Purity and Model Accuracy:** Imperfect cluster purity did not always hinder performance, as seen with MNIST, but further investigation is required to understand when and why purity gaps affect outcomes.

6.2 Future Work

These results validate the potential of clustering techniques to enhance federated learning, particularly in heterogeneous data environments. Moving forward, future work should focus on:

- Quantifying the impact of over-clustering, where excessive subdivision could reduce performance.
- Extending experiments to more diverse datasets and real-world applications, such as healthcare or autonomous systems.

- Investigating why specific datasets perform better under certain types of heterogeneity, such as label skew or domain shifts.
- Exploring advanced clustering techniques or hybrid methods to improve cluster purity and scalability.

Overall, this work lays a foundation for advancing clustered federated learning, with promising avenues for practical deployment in decentralized systems.

References

[1] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. 2020. Personalized Federated Learning with Theoretical Guarantees: A Model-Agnostic Meta-Learning Approach. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 3557–3568. https://proceedings.neurips.cc/paper_files/paper/2020/file/24389bfe4fe2eba8bf9aa9203a44cdad-Paper.pdf

[2] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. 2020. An Efficient Framework for Clustered Federated Learning. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 19586–19597. https://proceedings.neurips.cc/paper_files/paper/2020/file/e32cc80bf07915058ce90722ee17bb71-Paper.pdf

[3] Avishek Ghosh, Justin Hong, Dong Yin, and Kannan Ramchandran. 2019. Robust Federated Learning in a Heterogeneous Environment. *CoRR* abs/1906.06629 (2019). arXiv:1906.06629 http://arxiv.org/abs/1906.06629

[4] Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip Gibbons. 2020. The Non-IID Data Quagmire of Decentralized Machine Learning. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*, Hal Daumé III and Aarti Singh (Eds.). PMLR, 4387–4398. https://proceedings.mlr.press/v119/hsieh20a.html

[5] Paul Pu Liang, Terrance Liu, Ziyin Liu, Ruslan Salakhutdinov, and Louis-Philippe Morency. 2020. Think Locally, Act Globally: Federated Learning with Local and Global Representations. *CoRR* abs/2001.01523 (2020). arXiv:2001.01523 http://arxiv.org/abs/2001.01523

[6] Guodong Long, Ming Xie, Tao Shen, Tianyi Zhou, Xianzhi Wang, and Jing Jiang. 2023. Multi-center federated learning: clients clustering for better personalization. *World Wide Web* 26, 1 (2023), 481–500. https://doi.org/10.1007/s11280-022-01046-x

[7] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 54)*, Aarti Singh and Jerry Zhu (Eds.). PMLR, 1273–1282. https://proceedings.mlr.press/v54/mcmahan17a.html

[8] Yichen Ruan and Carlee Joe-Wong. 2022. FedSoft: Soft Clustered Federated Learning with Proximal Local Updating. *Proceedings of the AAAI Conference on Artificial Intelligence* 36, 7 (Jun. 2022), 8124–8131. https://doi.org/10.1609/aaai.v36i7.20785

[9] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. 2020. Robust and Communication-Efficient Federated Learning From Non-i.i.d. Data. *IEEE Transactions on Neural Networks and Learning Systems* 31, 9 (2020), 3400–3413. https://doi.org/10.1109/TNNLS.2019.2944481

[10] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9, 86 (2008), 2579–2605. http://jmlr.org/papers/v9/vandermaaten08a.html

[11] Kangkang Wang, Rajiv Mathews, Chloé Kiddon, Hubert Eichner, Françoise Beaufays, and Daniel Ramage. 2019. Federated Evaluation of On-device Personalization. arXiv:1910.10252 [cs.LG] https://arxiv.org/abs/1910.10252

[12] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. 2018. Federated Learning with Non-IID Data. (2018). https://doi.org/10.48550/ARXIV.1806.00582

[13] Hangyu Zhu, Jinjin Xu, Shiqing Liu, and Yaochu Jin. 2021. Federated learning on non-IID data: A survey. *Neurocomputing* 465 (2021), 371–390. https://doi.org/10.1016/j.neucom.2021.07.098

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009